Metrol. Meas. Syst., Vol. XIX (2012), No. 4, pp. 777-786.

METROLOGY AND MEASUREMENT SYSTEMS

Index 330930, ISSN 0860-8229

www.metrology.pg.gda.pl

RAPID PROTOTYPING METHODOLOGY OF EMBEDDED CONTROL-ACQUISITION SYSTEM

Jacek Augustyn, Andrzej Bień

AGH University of Science and Technology, Faculty of Electrical Engineering, Automatics, Computer Science and Biomedical Engineering, Al. Mickiewicza 30, 30-059 Kraków, Poland (✉ jacek.augustyn@agh.edu.pl, +48 12 617 4014, abien@agh.edu.pl)

Abstract

Modern control and measurement systems are equipped with interfaces to operate in local area networks and are typically intended to perform complicated data processing and control algorithms. The authors propose a digital system for rapid prototyping of target application devices. The concept solution separates the processing and control section from the hardware interface and user interface section. Both sections constitute independent ARM-based controllers interconnected via a direct USB link. Popular libraries can be used and low-level procedures developed, which enhances the system's economic viability. A test unit developed for the purpose of the study was built around a SoC ARM7 microsystem and an off-the-shelf palmtop device. It demonstrated a continuous data stream transfer capability up to 150 kB per second, which was sufficient to monitor the performance of an electricity line.

Keywords: embedded systems, control and measurement systems, hard real time systems, USB interface.

1. Introduction

Modern automated control and measurement systems demand high processing power and extensive programmer and user interfaces. As algorithms are becoming ever more sophisticated and complicated, the need for a simple and flexible user interface increases the challenge of producing flexible solutions that would be adaptable to individual end-user requirements. The proposed concept meets this challenge with a method of rapid prototyping, building and further developing an original solution.

Building control, diagnostic and measurement equipment is a multi-aspect process. The involved functionalities combine aspects of automation, metrology, electronics, as well as two areas of computer science, *i.e.* programming and system integration. While hardware modularity (ranging from integrated circuits to full modules) helps in achieving rapid rates of equipment prototyping (design and build), the kind of software required to achieve desired functionalities takes the bulk of the project effort.

In the opinion of the authors most of the project time is devoted to computer-science type of tasks that typically involve organising the transfer of measurement and control data while meeting hard real-time constraints. Computer literature reveals that for years the key factor influencing system lead-time and costs has been programmer productivity [1-2]. In the class of systems considered in this paper productivity pressures are focused on the development of systemic solutions, programming, multiple testing and patching. Indeed, it is the iterative nature of the designing and programming process [3] that delays its final completion. In the context of control and measurement devices the main problem is the multiple level subsystem integration (*e.g.* converter-processor, processor-to-computer, *etc.*). The most time-consuming tasks include building of a user interface to the required clarity and ergonomic criteria.

Article history: received on Mar. 7, 2012; accepted on Oct. 8, 2012; available online on Dec. 15, 2012; DOI: 10.2478/v10178-012-0069-9.

Commercially available libraries can help mitigate these problems, but at a considerable cost penalty and often restricting the system flexibility. An additional complication in the diagnostic and control equipment is involved in the demands of hard real-time constraints often requiring very short response times, *e.g.* below 1ms. The addition of measurement and recording functionalities also require the maintenance of a continuous data stream without losses.

Rapid prototyping tools, such as Matlab and LabView, can shorten the lead-time in delivering the functional code, but they are designed primarily to test the functionality of mathematical algorithms which constitute a very small percentage of the final source code. These tools are designed for very powerful processing platforms unsuitable for high-volume production. Additionally, power consumption is becoming a serious consideration owing to an overall drive towards environmentally friendly solutions. This puts a limit on the applicability of the standard power-hungry PC while alternative platforms require the bulk of the code to be dedicated to specific hardware, communications and user interface.

Designing and developing advanced solutions for measurement, diagnostic, recording and control systems intended for flexible low-budget production requires all these considerations to be taken into account at the prototyping and production stages.

There are a number of approaches to designing and building measurement, control and recording systems. Single-processor solutions tend to be the most popular, including particularly 8-bit microcontrollers [4-5] supported by additional controllers integrating with other systems, such as external data recording and storage. [6] proposes a hardware solution involving a combination of an ARM9 processor and the Linux system to handle a relatively slow fermentation process. Used are also specialized platforms as PC104 [7].Characteristics of other operational systems and their resource requirements are detailed in [8]. Other solutions involve mixed systems with 16-bit microcontrollers for the measurement and control functionality supported by a 32-bit processor [9]. Implementation of these concepts, however, requires adapting the Linux system to the target hardware platform, which is not only time consuming, but also requires specialized knowledge. Another suggested alternative is building proprietary hardware and software platforms [10] communicating via a local UART interface. This solution offers the advantage of a much better utilization of the hardware resources, but the implementation of own GUI libraries increases considerably the prototyping time because it requires a serious amount of programming. There is also a concept involving porting PC-computing environments into an embedded system [11]. This solution offers a control loop response time of 0.1 s. Finally, there exist off-the-shelf solutions based on typical mobile devices with specialized measurement modules [12-14], but their creators tend to limit the access to these solutions. These solutions use commercial libraries, which reduces the programming time, but adds to the overall cost and offers low bandwidth for continuous data streaming.

The authors propose a concept of dividing the measurement, diagnostic and control equipment to be implemented in separate hardware subsystems, which can be built with commercial-off-the-shelf components (COTS) and programmed using standard libraries and standard programming environments available for a given operational system. It is further proposed that strictly dedicated acquisition and execution sections ought to be implemented without an operating system to meet hard real-time constraints and to ensure short response times.

A direct USB connection between subsystems is proposed without additional converters.

2. Proposed solution

The proposed system comprises three main subsystems covering the measurement, diagnostic and control functions:

– A measurement and executive subsystem, including analogue circuits and transducers;
– A management subsystem that can additionally perform processing, diagnostic, recording and control functions; and
– A user interface and visualisation subsystem (*e.g.* keyboard and display).

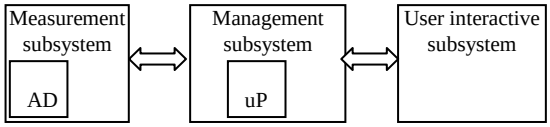

Fig. 1. Concept flow diagram of a control/measurement and diagnostic system.

Each of these subsystems is essentially a complex component itself containing another layer of abstraction with cooperating elements.

The paper proposes a modular solution with standardized communications to lighten the programming effort and subsystem integration. This is one of our main assumptions accommodating the concept of using standard programming libraries to mitigate the programmer productivity constraint. The proposed general architecture is illustrated on Fig. 2.

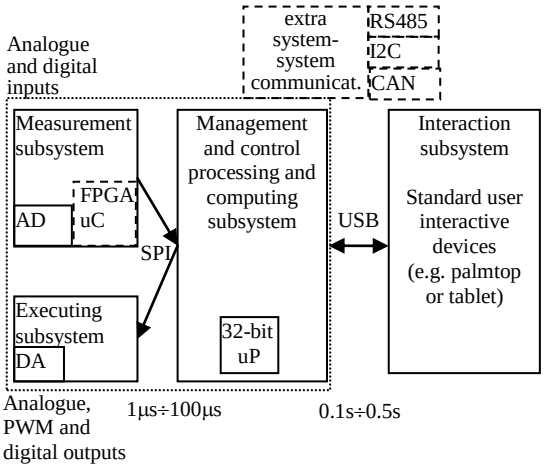

Fig. 2. Proposed system structure.

This illustrates the breakdown into: 1) the analogue subsystem strictly dedicated to the application and comprising inputs and outputs and voltage adapting circuits; 2) the management subsystem; and 3) the standard user interaction subsystem. The desired communication times between sections ranged between 1÷100 µs and 0.1÷0.5 s, respectively.

2.1. Proposed hardware solution

It is proposed to break down the entire system into two physical parts: 1) one that is based on an SoC (system on chip) class microsystem and running software without an operating system; and 2) an off-the-shelf part using a standard operating system and a user interface. While this may seem to increase hardware complexity, it only entails designing and building dedicated analogue circuitry (for example there is no need for broadband data buses or multi-layered printing). The software-side advantages are detailed in Section 2.3.

The management and visualization-archiving subsystems are proposed to be interconnected via a direct USB interface. This offers a net transfer speed in the order of 600 kB/s [15], which is sufficient to record a continuous data stream around 200 kSPS. The USB interface is an inexpensive and popular solution and complete modules are available in the microsystems proposed below.

In the management subsystem it is proposed to use 32-bit SoC (system-on-chip) processors, either the ubiquitous ARM, such as SAM7 [16], or the CortexM, *e.g.* SAM3 [17], which has been recently gaining in popularity. They offer several dozen MIPS (million instructions per second), which is sufficient for executing auxiliary processing procedures (*e.g.* filtering, computing effective values, RMS and correlations). The calculations can be performed with 32-bit precision. These SoCs have a low price and offer sufficient RAM capacities to buffer data received from a transducer and in this way they reduce the required frequency of communication. Importantly the processors feature a vast set of peripherals (including a multiple SPI port, a full-speed hardware USB-UDP port and UART and I2C ports, ADC, DAC, as well as CAN, EthernetMAC in SAM7X family). Standing out from the field are the SAM7/SAM3 families equipped with the peripheral DMA controller (DMA-PDC) providing an individual DMA channel for each piece of fast peripheral hardware, thus avoiding channel conflicts between different peripherals. With right programming of the DMA controller the workload on the processor core can be reduced considerably and skilful use of hardware buffer switching reduces the risk of breaking the data stream continuity [18].

The chips can be programmed in C using free programming tools. In a non-OS configuration [18] it is possible to achieve fast and reliable response times in the order of 1÷100 μs to external event. The processor has sufficient power to perform control algorithms (such as PID, LQG, neural control, fuzzy control [19]), as well as measurement and communication tasks [20-21].

The processors work well with external FLASH chips and SD cards. Large memory capabilities can be used to store vast database files of registered results. One drawback with SD cards is that the simplicity of their handing algorithms may cause blocking of the data subsystem or a considerable degradation of the system's processing power. Writing to FLASH or SD memory may involve combing through a large number of sectors in the allocation tables before an available sector is found and leading not only to relatively long write times, but also to low predictabilities of the file response times [22].

SPI/FastSPI interfaces are proposed for standardization of the communication between the measurement/executive and the management part. This is a relatively easily programmable interface offering speeds of dozens Mb/s and allowing the required data exchange times in the order of 1÷100 μs. Transducer vendors offer such types of SPI variants, which standardizes this part of the subsystem.

Where extended processing is needed an additional FPGA circuit or an additional microcontroller can be used. Normally, substantive algorithm kernels, such as FIR filtration, IIR filtration, correlation, FFT and matrix multiplications are standard and replicable. They also rarely pose implementation problems, since open source libraries and source codes are available. Most of the implementation time is devoted to programming and testing communication algorithms and subsystem operating synchronization. The key issue here is the aforementioned programmer productivity, which covers also VHDL programming. Adoption of an SPI interface on both sides of the additional processing FPGA or microcontroller speeds up the development of the system and the proposed "SPI in/SPI out" architecture simplifies the overall design and reduces cost. An alternative solution would be to use the SSI (I2S) standard found on many types of transducers and microsystems and is relatively easy to implement in FPGA circuits.

A one-chip microsystem can handle all the measurement, executive and management subsystem as long as it has embedded transducers meeting the required metrological parameters.

The system can be expanded by adding I2C, UART/RS485 and CAN interfaces. An ARM microcontroller can also communicate with asynchronous serial ports at transfer speeds of several Mbps, which makes it easy to expand the system with additional – *e.g.* 8-bit – microcontrollers. The more advanced versions can include Ethernet handling in the measurement and execution parts. For example the SAM7X family has a built-in EMAC port and only requires an addition of a PHY voltage converter on the outside.

2.2. Proposed software solution – the management part

The management part, later referred to as the application, is proposed to be a non-OS implementation. The key advantage of this approach involves relatively easily achievable fulfilment of hard real time constraints within the adopted time range of 1÷100 μs. The lack of an OS simplifies the programming effort, because there is no OS for the programmers to learn about their low-level details nor is there any need to code hardware drivers cooperating with an OS. The required 1÷100 μs response time means that the use of an operating system or kernel would run into serious difficulties. Indeed, while it is possible for embedded real-time systems to achieve thread switching times of 30 μs [23], in practice this is limited to a single thread with the highest momentary priority and with a powerful processor. In general, due to the multiple processes and threads and their mutual relationships the worst response time may be in the order of 10ms. This would make fast measurement and control systems difficult to achieve.

It is true that source codes of many systems, such as Linux, are available, but their suitable modification requires programmers to know its specificity and kernel. An OS also requires implementing low-latency hardware drivers and an architectural adaptation to the OS. Implementation of controllers to guarantee response times in the order of 1÷100 μs is very time-consuming both in terms of learning the design of OS modules and their interconnections and then their integration with the system and subsequently testing and patching.

Non-OS software reduces these problems and makes it much easier to achieve the assumed response times. There is no requirement to know details of the system interaction and any external hardware can be handled by a short-sequence code. Potential drawbacks, such as a lack of tools to handle the user interface and the file system are irrelevant as these will not be used in this part of the proposed solution.

The development of the management subsystem software involves multiple loading of the code into the FLASH memory. This is necessary due to the iterative nature of the programming process despite relatively long loading times with applications containing 100-200kB of resultant code (for example the execution of the trivial *printf()* function for the GNU C compiler takes ca. 40 kB of code). It is the FLASH memory that is the bottleneck in application development, as it only allows setting up 2-4 hardware traps (depending on the JTAG interface and target processor capacities) thus limiting the debugging potential. RAM allows unlimited traps and clearly improves debugging potential, but SoCs have insufficient RAM resources to load and run the entire programs.

In order to make application development easier it is proposed to build an embedded library with a set of typical procedures, such as character formatting handling (*e.g. sprintf()*), optimized hardware data exchange, start-up codes and non-blocking functions allowing character-based output. This addresses the problem of the blocking nature of the standard *printf()*, which requires waiting for the completion of character output thus distorting the time

predictability and making the standard *printf()* unsuitable for fast systems. A set of character output functions should replace the *printf*() that comes with the compiler and should allow character output in a deterministic non-blocking way. The USB interface is not recommended for this purpose because each reloading of an application results in a PC-side driver reload leaving the connection blocked for up to 50ms [15]. For this reason a UART port is proposed instead. Similarly the library should allow input of character commands in the application without blocking the main processing. Adopting a non-OS solution makes a library like that easier to implement with just the requirement of technical ability to handle the compiler and linker on the target platform. The library could include a set of framework drivers for the SPI port with the streaming performance in the order of 500-1000kSPS and optimized to minimize the microsystem core loading to single percentage points. Relevant guidelines can be found in [18]. While microsystem vendors publish framework command sequences for port configuration, the source codes are not optimized for time performance. This framework illustrates configuration sequences and typically represents the blocking versions. In developing a library the drivers ought to be optimized using for example DMA channels and an interrupt controller so that they can be implemented in a non-blocking way. An embedded USB handling feature is also very convenient. Memory and processing power requirements are detailed in [15]. The library should also include procedures for a fast native multi-thread kernel with response time in the order of microseconds. Due to the proposed non-OS approach the library can be developed in parallel with other efforts.

One of the more important advantages of the proposed approach is the ready-made application framework, which offers the executor of the measurement and management components, containing the *main*() function and is so similar to a classical C language programme that substantive work can start almost immediately. The software developer needs only a general programming knowledge. He can use a substituted non-blocking *printf()* function in a standard manner and observe the effect of the application's operation on a terminal in real time. Working with a terminal is sufficient for the development stage and allows the system-user interaction of the management system (setting parameters, modes and displaying results) without prior implementation of a GUI. This helps in parallel project development.

Another advantage of the proposed solution is that a relatively large, but rarely modified, code can be separated and placed permanently in the FLASH memory as an embedded core. The application framework only calls the embedded functions and can therefore be very small in size. Initially, it indeed is only single KBs that can load very fast to target SoC and run from the microsystem's RAM for debugging purposes without limits on the number of traps. Successfully tested sections can then be moved to the embedded part. Using the RAM for this process has an additional advantage of saving the limited FLASH memory write cycles.

2.3. Proposed visualization and recording part

The provision of adequate end-user ergonomics and aesthetics is a time-consuming task. Designing, engineering, building and providing software for specialized hardware poses additional difficulties in the hardware (*e.g.* mechanical construction) and software (handling of the display, graphics, fonts) areas, which extends the prototyping phase.

To address this concern it is proposed to implement the user interaction and recording system on an commercial off-the-shelf (COTS) mobile device, such as a palmtop, tablet, smartphone, or even a car sat-nav system running an accessible OS. These mass-produced devices feature low unit costs, compact design and color touchscreens as standard. Programming libraries, including graphic libraries, are available for their operating systems. Designed to be portable they have many features of heavy-duty equipment, such as improved

shock and vibration resistance and temperature tolerances greater than regular desktop devices. Importantly, they are energy efficient and battery powered, which solves the issue of buffer power supply in case of mains power outages making them suitable as portable measurement equipment. The devices can handle large volumes of mass-storage (SD card or even HDD disk). In view of this range of advantages, building of a proprietary solution for this section must be seen as non-viable.

The rates of data exchange between this section and the rest of the system in the overall system architecture are assumed within the range of 0.1÷0.5 s, which opens up the possibility to use a standard WindowsCE/Mobile or Android solution. This offers an important advantage of allowing much of the programming effort to be entrusted to developers with just a general programming knowledge, thus considerably reducing the lead time. The operating systems offer functions and libraries that can be used to build GUIs with extensive structures and capabilities and for data recording and presentation systems.

The proposed solution has the considerable advantage of an expected forward compatibility, as hardware and software vendors are bound to maintain compatibility of their future products with the current systems for years to come. Solutions developed in accordance with the proposed approach will have long lives in terms of ongoing development, maintenance and availability of replacement units. Typical industrial life spans of 5-10 years will be easily achievable. Future versions of current standards, such as Bluetooth, WiFi or SDHC, will be relatively easily integrated by using standardized programming libraries. Thus proposed, solution can be easy adopted to realisation of smart-grid metering [29].

The only requirement for specialized programming knowledge is limited to the handling of data exchange between the user interaction section and the measurement and control parts. Commonly, however, off-the-shelf devices are closed proprietary designs, their manufacturers restrict access to detailed drawings and learning details of their internal connections would be excessively time-consuming. It is proposed to use a USB connector with a host. It offers much greater speeds than the standard RS232 connector and is present on most mobile devices unlike the RS232. Normally an USB host is integrated with the main processor, *e.g.* [24], which reduces the overall cost, but makes programming much more complicated than in the case of a serial UART port. One solution is to use commercial libraries, such as LabView or [25], but this drives the cost up. However, the communication times adopted in the proposed solution (0.1÷0.5 s) are low enough to use devices with standard USB classes for which free drivers are available and purchase of commercial modules can be avoided. The low transfer times offer the added benefit of avoiding the necessity of deep intervention in the operating system and driver codes.

The visualization and logging section can utilize standard programming libraries normally available with the operating system. The communication times assumed for the solution make it particularly easy to use typical file system handling libraries.

3. Practical implementation

An example of the proposed solution was built using a SAM7X family microsystem in the management, measurement, executing section and a palmtop device in the visualization-logging section (Fig. 3). The system was designed to measure and record effective voltage values. It handles 8 measurement channels, 16 inputs and 4 PWM output channels and 16 discrete outputs. The management section can accommodate regulating algorithms or additional diagnostic algorithms.

The microsystem was programmed using a free GNU C compiler. The total source code had approximately 12000 lines that yielded 110 kB of binary code. 90% of the code was

separated and placed as an embedded library used as a universal resource containing start-up codes, modules supporting the SPI connector and converters, a basic interaction module for the RS232 and terminal character and a USB support module. This embedded library can be used for many other projects.

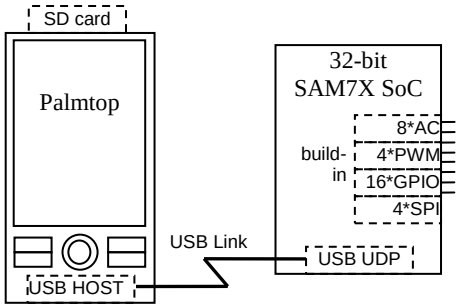

Fig. 3. Implemented system.

The proposed system architecture worked well with the WindowsMobile system running on the palmtop without modification. The visualisation and recording section was programmed using the VisualStudio environment, which was much cheaper than commercial library packages, such as LabView or Matlab. A 90-day trial version of the package proved sufficient for the test implementation and the license must only be purchased when the solution is commercialized. A mixed programming approach was used. The GUI was built with a graphical application builder and the .NET language to shorten the lead-time. The communication function was implemented in C/C++, which – as a lower level language compared to .NET – produced a faster output code.

In the visualization and recording section only the USB area required specialized knowledge, while the remaining part was implemented by programmers with just a general programming knowledge, as long as they knew the selected VisualStudio environment.

The most difficult part was integrating the subsystems via the USB interface. A CDC class solution was finally selected for data exchange [26] due to its better data stream performance than the popular HID class [27]. Standard programming components were used throughout. During the implementation stage, certain code modifications were necessary both on the WindowsMobile system and the SAM7X microsystem side due to implementation differences.

This implementation of bidirectional communication allows manipulation of system operation parameters. The acquired data can be recorded on an SD card.

The use of standard components facilitated a continuous data stream in the order of 30÷150 kB/s. The actual performance depended on the size of packages, organization of their transfer and system parameters. The fastest transfers were achieved with 4 kB packages, but their further enlargement produced no transfer gains. Testing showed that incorrect parameter values could limit transfer speeds to 64 kB/s.

These results were comparable to the performance achieved with commercial RS-USB converters (like FTDI chips). The 3Mbit/s transfers reported for these converters only pertain to hardware solutions used in the component, which means that in practice the figure can be much lower and highly dependent on the implementation of the software driver.

The observed data exchange time between subsystems ranged widely from 15 ms to 300 ms. These differences were due to different implementations of the various software branches. The maximum values were observed during screen refreshing and writing data to SD media and the frequency of their occurrence was strictly liked to the frequency of refreshing.

4. Conclusion

A measurement and recording system was implemented with a SoC microsystem and a visualization-recording part based on an commercially off-the-shelf (COTS) mobile device. The system is used to measure parameters of electricity. The proposed concept involving standard hardware and software components helped shorten the prototyping phase and considerably reduced its cost. It allows fast prototyping of measurement, control and recording equipment for small-scale production with a potential for adaptation to suit the end-user. The broad use of standard components and solutions reduced the implementation costs and improved the system maintainability and potential for expansion.

The major difficulty was encountered when programming the USB connection and ensuring adequate time synchronization. It was demonstrated, however, that it was possible to achieve a continuous data stream at 150 kB/s with standard hardware and software components. In this way it is possible to build a system that records, for example, 6 analogue parameters with 12 bit resolution and 25 kHz frequency. Further optimization of the stream parameters is possible, but would require a deeper knowledge of the design and functioning of USB software drivers in a given hardware platform. This will be the subject of further investigation.

The concept offers an important advantage of lowering the required programmer skill levels. With the exception of the specialized communication module the programmers only needed general programming knowledge rather than specialized hardware and operating system expertise.

The concept discussed in this paper was implemented on a palmtop device and is generally easily transferable to other off-the-shelf hardware platforms with operating systems Windows CE such as [28]. It can also be easily adapted to Linux/Embedded-Linux and to the ever more popular Android system. The software implementation would clearly require different source codes, but the necessary standard components are available in these operation systems. The use of standard hardware solutions derived from smartphones running on Android seems to show particular promise.

References

[1] Gamma E., Helm, R. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison Wesley.

[2] Romeo, F. (2001). Embedded Systems: the Real Story. *38th Design Automation Conference, Plenary Panel*. Las Vegas.

[3] Wrycza, S., Marcinkowski, B., Wyrzykowski K. (2006). *UML 2.0 in modelling of information systems.* Gliwice, Helion, 456. (in Polish)

[4] Bulbenkiene, V., Pecko, A., Zulkas, E., Kuprinavicius, A., Sokolov, A., Mumgaudis, G. Energy (2011). Sub-Metering Data Acquisition System. *Electronics and electrical engineering*, 5, 99-102.

[5] Chan, S., Teng, J., Chen, C., Chang, D. (2010). Multi-functional power quality monitoring and report-back system. *Electrical Power and Energy Systems*, 32, 728–735.

[6] Zhang, L., Wang, Z. (2010). Design of Embedded Control System Based on ARM9 Microcontroller. *International Conference on Electrical and Control Engineering*, 3579-3582.

[7] Szrek, J., Wójtowicz, P. (2010). Idea of wheel-legged robot and its control system design. *Bull. Pol. Acad. Sci.-Tech. Sci.*, 58(1), 43-50.

[8] Cho, M.H., Lee, Ch.H. (2010). Low-Power Real-Time Operating System for ARC (Actual Remote Control) Wearable Device. *IEEE Transactions on Consumer Electronics*, 56(3), 1602-6909.

[9] Simarro, R., Coronel, J., Simo, J., Blanes, J.F. (2008). Hierarchical and Distributed Embedded Control Kernel. *17th IFAC World Congress*, Coex.

[10] Yang, J., Gao, Z., Tang, J., Chen, Y. (2010). A Practical Predictive Control Algorithm for Embedded System and Its Application. *Journal of Computational Information Systems*, 6(11), 3501-3508.

[11] Ma, L., Xia, F., Peng, Z. (2008). Integrated Design and Implementation of Embedded Control Systems with Scilab. *Sensors*, 5501-5515.

[12] Precision USB thermometer probe. (2010). Electronic Temperature Instruments Ltd., *http://www.etiltd.com*

[13] Three-axis Hall Magnetometer THM1176 Users Manual. (2008). Metrolab Instruments SA.

[14] *I-Scan*® Handheld Pressure Measurement System. (2008). Tekscan, Inc. *http://www.tekscan.com/industrial /iscan-handheld.html.*

[15] Augustyn, J., Bień, A. (2009). Real time performance of USB interface in embedded control and measurement systems. *Przegląd Elektrotechniczny*, 7, 1-7. (in Polish)

[16] AT91SAM ARM-based FLASH MCU Doc. 6120I. (2011). Atmel.

[17] AT91SAM ARM-based FLASH MCU SAM3S Series Doc. 6500C. (2011). Atmel.

[18] Augustyn, J. (2007). *Design of embedded systems with application to SAM7S family with ARM7TDMI core*. Kraków, Wydawnictwo IGSMiE PAN, 302. (in Polish)

[19] Franc, H., Šafari, R. (2010). ARM-Cortex Microcontroller fuzzy position control on an automatic door test-bed. *19th International Workshop on Robotics in Alpe-Adria-Danube Region – RAAD*, Budapest, 417-422.

[20] Pal, T., Shekhar, Ch., Sharma, H.D. (2009). Design and Implementation of Embedded Speed Controller on ARM for Micromanufacturing Applications. *International Conference on Advances in Computing, Control and Telecommunication Technologies*, 406-410.

[21] Jianmin, H., Kai, G., Hong, Ch., Na, R. (2007). Design of Micro-oxidation power control system based on LPC2119. *The Eighth International Conference on Electronic Measurement and Instruments ICEMI*, Xian, 849-853.

[22] FAT: General Overview of On-Disk Format. Version 1.03. (2000). Microsoft Corporation.

[23] Microsoft Windows Embedded CE 6.0 Intel Atom Processor. (2009). Intel.

[24] Intel® PXA27x Processor Family Developer's Manual. (2004). Intel.

[25] USB CDC/ACM Class Driver for Windows CE, Reference Manual, ver 1.4. (2009). Thesycon Systemsoftware & Consulting GmbH.

[26] Universal Serial Bus Class Definitions for Communications Devices. Revision 1.2. (Nov. 2007). *www.usb.org*

[27] Device Class Definition for Human Interface Devices (HID), Version 1.11. (2001). *www.usb.org*

[28] Mini 6410 Hardware Spec, Rev 0.1.0. (Mar. 2011). Guangzho. *www.friendlyarm.net*

[29] Benysek, G., Kazmierkowski, M.P., Popczy., J., Strzelecki, R. (2011). Power electronic systems as a crucial part of Smart Grid infrastructure – a survey. *Bull. Pol. Acad. Sci.-Tech. Sci.*, 59(4), 455-473.